Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en
Ciencia e Ingeniería Computacional

Modelo de red de interconexión basado en enlace

Memoria del trabajo de ”Iniciación a la Investigación. Trabajo de Fin de Máster” del “Máster en Ciencia e Ingeniería Computacional”, realizada por Alex Ramón Gómez Vera, bajo la dirección de Daniel Franco Puentes Presentada en la Escuela de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

2010

Iniciación a la investigación. Trabajo de fin de máster

Máster en Ciencia e Ingeniería Computacional.

Título
Modelo de red de interconexión basado en enlace

Realizada por Alex Ramón Gómez Vera en la Escuela de Ingeniería, en el Departamento Arquitectura de Computadores y Sistemas Operativos.

Dirigida por: Dr. Daniel Franco

Firmado

Director Estudiante

Resumen

La computación de altas prestaciones es una área de la informática que evoluciona rápidamente, en la que actualmente aparecen nuevos computadores que llegan a los petaflops.
Al principio del trabajo, se estudian los distintos tipos de redes de interconexión y los modelos de red que se utilizan para medir su latencia.
El objetivo de este trabajo, es el diseño, implementación y simulación de un modelo de red de interconexión basado en enlace, que tiene en cuenta la información de topología y enrutamiento de la red de interconexión. Teniendo en cuenta que los modelos son una abstracción del sistema, en éste trabajo se hace la verificación y validación del modelo, para asegurar que éste se aproxima a lo planteado en el diseño y también que se parece al sistema que se quiere modelar.

Palabras Clave: Redes de interconexión, Modelos de red, Simulación, Implementación.

Resum

La computació d'altes prestacions és una àrea de la informàtica que evoluciona ràpidament, on actualment apareixen nous ordinadors que arriben als Petaflops.
Al principi del treball, s'estudien els diferents tipus de xarxes d'interconnexió i els models de xarxa que s'utilitzen per mesurar la seva latència.
L'objectiu d'aquest treball, és el disseny, implementació i simulació d'un model de xarxa d'interconnexió basat en enllaç, que té en compte la informació de topologia i enrutament de la xarxa d'interconnexió. Tenint en compte que els models són una abstracció del sistema, en aquest treball es fa la verificació i validació del model, per assegurar que aquest s'aproxima al plantejat en el disseny i també que s'assembla al sistema que es vol modelar.

Paraules claus: Xarxes d'interconnexió, Models de xarxa, Simulació, Implementació.

Abstract

The high-performance computing is an area of rapidly evolving in computer technology, in which new computers are now coming to petaflops.
At the beginning of work, we study the different types of interconnection networks and network models that are used to measure the latency.
The aim of this work is the design, implementation and simulation of a link-based interconnection network model, which takes into account the topology information and routing of the interconnection network. Given that the models are an abstraction of the system, in this work we has to verify and validate the model, to ensure that it is close to the proposed design and that looks like the system to be modeled.

Keywords: interconnection networks, network models, simulation, implementation.

Al profesor Daniel Franco,
por su gran colaboración y amabilidad.

A Diego y el grupo de investigación,
por su asesoría y buenos aportes.

A Jennifer,
por su amor y paciencia.

A mi familia,
por el apoyo incondicional.

Índice general

Índice de figuras

Capítulo 1

Introducción

El mundo industrial y académico actual está sediento de aumentar *FLOPS (Floating point Operations per Second)* [1].

Actualmente, existen aplicaciones que se consideran *Grand Challenge*, porque no pueden ser resueltas en un tiempo considerable, como ejemplo tenemos:

- Simulaciones de ecosistemas
- Dinámica de fluidos aplicada.
- Simulaciones de energía nuclear.
- Simulaciones para macroeconomía.
- Modelado del ambiente a Macro-escala.
- Biomecánica.

Cada día avanza la tecnología y por tanto nacen nuevas aplicaciones con grandes necesidades de cómputo.

Muchas de estas aplicaciones que hace apenas unos años, requerían varias semanas para ejecutarse, hoy lo hacen en pocas horas.

En las últimas décadas la velocidad de los procesadores ha ido en aumento cada año y el número de transistores contenido en un microprocesador ha ido duplicándose cada 18 meses [2], gracias a los avances en microelectrónica. Actualmente, las pistas o caminos entre los transistores tienen unos cuantos *nm* de

[1] Operaciones de punto flotante por segundo
[2] Ley de Moore

ancho y teniendo en cuenta que lo que circula por esas pistas son electrones, se podría llegar en un futuro a un punto en que sólo podría pasar un electrón al tiempo por un camino, lo cual es una barrera física clara. Sin embargo, como lo que se necesita es aumentar los *flops* para ejecutar más rápido las aplicaciones, por ello aparecen los computadores paralelos.
Se utilizan distintas arquitecturas de computadores paralelos entre las que tenemos:

1. Multicore computing
2. Symmetric multiprocesing
3. En Distributed computing tenemos
    - Massive Parallel computing
    - Cluster computing
    - Grid computing

En donde *Cluster* y *MPP (Massive Parallel Processing* son las más ampliamente utilizadas por los supercomputadores que están entre los 500 más rápidos.

El supercomputador más rápido actualmente es el *Jaguar* del NCCS, Centro Nacional de Ciencias de la computación en EEUU. Esta supermáquina llega a unos increíbles 1,75 Petaflops [2], teniendo 224162 *cores* y usando arquitectura MPP Cray XT5.

Para interconectar un computador paralelo con un numero de *cores* alto, se requiere una gran red de interconexión de alto rendimiento, que no genere mayores retardos, ya sea cluster o MPP.
Las redes de interconexión son una parte fundamental en los computadores paralelos masivos, por lo que se requiere modelar estas redes para poder hacer predicciones, simular cambios, y ahorrar costes.
Normalmente los sistemas complejos como los supercomputadores no pueden pararse para hacer pruebas de nuevos protocolos o nuevos cambios, porque el coste económico sería muy alto, no sólo en las horas de caída del sistema, sino en horas empleadas ante algún desastre que hayan producido las pruebas.

Es preferible tener un entorno controlado donde se pueda evaluar el comportamiento de los diversos elementos del sistema y se pueda por tanto predecir como

reaccionará ante los cambios planteados.

En general, algunas de las ventajas de modelar y simular serían las siguientes [3]:

1. No es necesario interrumpir las operaciones de la compañia.
2. Proporciona muchos tipos de alternativas posibles de explorar.
3. Proporciona un método más simple de solución cuando los procedimientos matemáticos son complejos y difíciles.
4. Proporciona un control total sobre el tiempo, debido a que un fenómeno se puede acelerar.
5. Auxilia el proceso de innovación ya que permite al experimentador observar y jugar con el sistema.
6. Generalmente es más barato mejorar el sistema vía simulación que hacerlo en el sistema real.
7. Es mucho más sencillo visualizar y comprender los métodos de simulación que los métodos puramente analíticos.
8. Da un entendimiento profundo del sistema.

1.1. Problema de los modelos de red de interconexión

Los modelos son una abstracción del sistema real, por lo que tienen en cuenta sólo parte de la informaciónHacer experimentos para comparar la velocidad y precisión del modelo, respecto a a un modelo de referencia. para explicar el fenómeno investigado o dicho de otra manera, es una simplificación que nos ayuda a mejorar el conocimiento de sistema que se estudia.

Las redes de interconexión son un sistema bastante complejo, por lo que intentar tener todas las variables, sucesos, fallos en un momento dado no ayudaría a comprender el fenómeno que se quiere estudiar.

[3]Tomado de la clase Modelado y Simulación, CSE 2008-2009

Los modelos de red simplifican o abstraen el comportamiento de una red, que según cual, puede tener en cuenta distintos componentes. Si el modelo es muy general, en la validación [4] puede tener un error muy alto. Si el modelo es muy detallado, y tiene en cuenta la gran mayoría de los componentes de la red de interconexión, la complejidad dificultaría la comprensión del modelo y la implementación en un entorno computacional.
Al diseñar un modelo de red de interconexión se debe especificar que componentes, y que sucesos son los que tendremos en cuenta, sin generalizar ni detallar al máximo.
Hay una amplia gama de modelos de redes de interconexión, hay una gran multitud de investigadores ideando e implementando cada día nuevos modelos.

Sin embargo, nos vamos a centrar en la tesis [6] donde se mencionan cuatro modelos de red de interconexión que por orden de complejidad, irían de la siguiente manera:

- Global–Step
- Global–Average
- Destination–Based
- Link–Based

Ante más complejidad, mayor precisión, pero también mayor lentitud. Los modelos *Global* consideran la red como una caja negra, por lo que dan un error $\approx 30\,\%$ aunque con un aumento de velocidad de 100 veces en promedio.
Por otro lado, el modelo basado en destino *(Destination–Based)* tiene en cuenta la información contenida en los paquetes entrantes, lo que le da una disminución en el error y de la velocidad.

1.1.1. Problema del modelo basado en destino *DBModel*

El modelo basado en destino hace un revisión de los paquetes entrantes, los clasifica por destino, va creando una *cola* distinta para cada destino, y hace un procedimiento recurrente para hallar la latencia de cada pareja *origen-destino* en un intervalo de tiempo que llamaremos *quantum*.
Por ejemplo, los distintos paquetes que tienen un destino X van a ser almacenados

[4] Se explicará más a fondo este término en un capítulo posterior

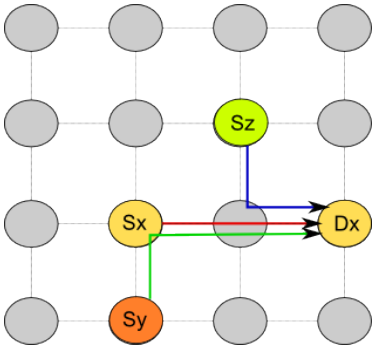


Figura 1.1: Choque al mismo destino

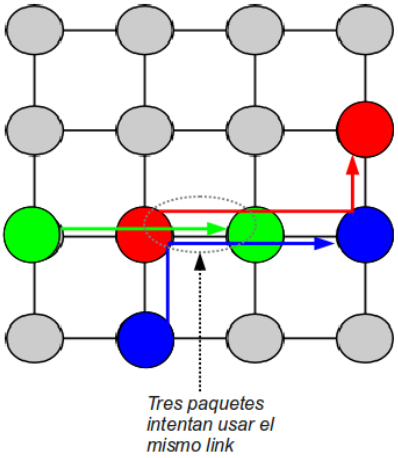


Figura 1.2: Choque al usar el mismo enlace

en una *cola* llamada *Destino X* y a estos paquetes se les calculará la latencia con un procedimiento recurrente que será explicado en detalle en un capitulo posterior.

El *DBModel* no tiene la información de topología y de enrutamiento de la red, es decir, no tiene en cuenta el camino que sigue el paquete y por lo tanto, no sabé que nodos o que enlaces pueden estar ocupados por otro paquete.

El error de precisión de éste modelo es $\approx 8\,\%$ comparando con un modelo de referencia hecho en el simulador Opnet [10]. La velocidad de simulación es ≈ 30 veces comparado con Opnet.

El modelo de red basado en destino *(DBModel)* detecta el caso de la figura 1.1 pero no detectaría un comportamiento como el que se plantea el la figura 1.2 cuando chocan varios paquetes que intentan usar el mismo link, esto implicaría claramente un error de precisión.

Como se quiere disminuir el error , el modelo basado en enlace detectará cuando choquen varios paquetes que intenten usar el mismo enlace aunque no tengan el mismo destino, lo que nos permitirá reducir el error de precisión sin sacrificar en gran medida la velocidad de simulación.

1.2. Objetivos

En vista que los modelos de interconexión juegan un papel importante en los computadores paralelos, y se requieren modelos que hagan predicciones de su rendimiento, por lo tanto en este memoria de investigación se procederá a:

> Diseñar e implementar un modelo de red de interconexión que tenga en cuenta la información de topología y enrutamiento, para poder detectar la contención que se producen cuando dos o más paquetes intentan usar el mismo enlace dentro de un *quantum* [5] de tiempo.

Cuando se ha construido e implementado el modelo, es necesario *verificar que la implementación este de acuerdo con lo diseñado, por lo tanto, se deben hacer experimentos para verificar la concordancia, y ver si el modelo esta haciendo lo que se esperaba.*

Una vez se ha verificado el modelo, se debe hacer la validación, es decir, se deben hacer experimentos precisión del modelo, respecto a un modelo de referencia.

Teniendo en cuenta que los modelos son simplificaciones del sistema y que en la mayoría de los casos tienen un error de precisión, en el modelo que se diseña se busca una compaginación entre velocidad y precisión.

[5] Es un valor dado por el usuario que se usa paralelos sincronizar los datos que se entregan al FSS

Capítulo 2

Marco Teórico

En este capítulo se explorará sobre algunos términos que son familiares, pero que normalmente se ignora su significado a profundidad. En la primera sección se trata sobre los sistemas, los modelos y simulación, en un sentido amplio.

En la segunda y tercera sección se realiza una breve exposición teórica de las redes de interconexión usadas en altas prestaciones y de algunos modelos de dichas redes.

2.1. Sistemas, Modelos y Simulación

2.1.1. Sistemas

Un *sistema* es definido como una colección de entidades que interactúan entre sí, para el logro de un determinado fin [5].

Las diferentes formas de estudiar un sistema [5] de la figura 2.1, nos permite abordar como se quiere experimentar. como se observa se debe crear un modelo matemático para proceder con la simulación.

En el segundo nivel de la figura 2.1, podemos escoger entre usar el sistema real o un modelo del sistema. La experimentación con el sistema real puede plantear problemas éticos o económicos [4]. En tales casos, puede procederse a construir una versión simplificada, o *prototipo* del sistema.

2.1.2. Modelos

Un modelo puede definirse como una representación simplificada de un sistema real o un proceso o una teoría, con el que se pretende aumentar su comprensión,

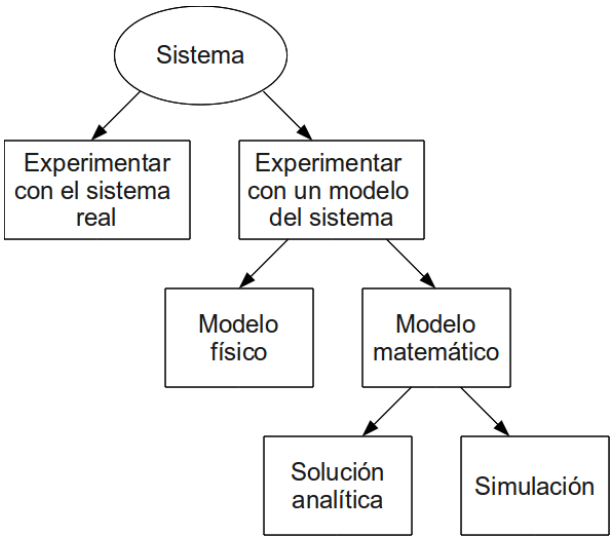


Figura 2.1: Formas de estudiar un sistema

hacer predicciones y, posiblemente, ayudar a controlar y mejorar el sistema [4, cap 5.].

Los modelos son una abstracción del *sistema* real para ayudar a entender como funciona. Por lo tanto, en un modelo siempre hay que tener en cuenta que no se tiene toda la información del sistema y que se puede perder precisión, aunque este bien planteado.

En el tercer nivel de la figura 2.1, podemos escoger entre usar un modelo físico o un modelo matemático. Un modelo físico es lo que se imagina la mayoría de personas, dos ejemplos de éstos son las camaras de viento y las cabinas de entrenamiento de pilotos. Estos modelos pueden ser muy útiles para unos cuantos casos contados. Pero, la gran mayoría de problemas se resuelven con modelos matemáticos. Por ejemplo, uno de los modelos matemáticos más sencillos es $d = v * t$, donde v es la velocidad y t es el tiempo empleado.

Mirando el cuarto nivel de la figura 2.1, se pueden escoger entre dos opciones, una solución analitica o con simulación.
Una vez se ha construido el modelo matemático, se debe examinar su complejidad. Si es baja puede tener una solución analítica, en caso contrario se debe hacer mediante simulación.

Se pueden hacer distintos modelos matemáticos de un mismo sistema variando en complejidad cada uno. Por ejemplo, tenemos una pelota caída libre; el modelo mas sencillo sólo tiene en cuenta la gravedad, pero otro modelo más complejo

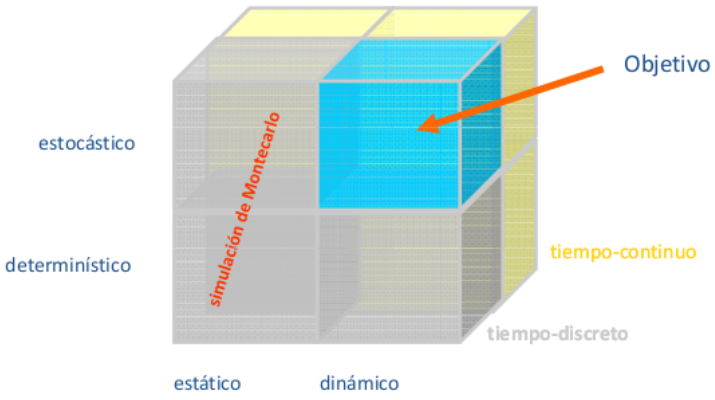


Figura 2.2: Modelos de simulación

tendría en cuenta adicionalmente la resistencia del aire, lo que le dará más precisión. Pero al aumentar la complejidad del modelo también se aumenta la dificultad de entenderlo y simularlo, al igual que aumentaría la velocidad de simulación.

El término modelo puede tener diferentes significados para distintas personas y la representación de un modelo puede ser distinta. Otra clasificación, considera esencialmente tres clases [4, cap.5] de modelos:

- *Modelos físicos:* Son representaciones de sistemas físicos y están descritos por variables medibles.
- *Modelos mentales:* Son modelos heuristicos o intuitivos que sólo existen en nuestras mentes. Son imprecisos, difusos y díficiles de comunicar.
- *Modelos simbólicos:* Son aquellos que incluyen operaciones lógicas o matemáticas que pueden utilizarse para formular una solución de un problema. En esta clasificacin están los modelos matemáticos.

2.1.3. Simulación

Si el modelo matemático debe ser estudiado mediante simulación, se puede escoger entre distintos tipos de modelos de simulación.

En la figura 2.2, se muestra los distintos tipos de modelos de simulación. La mitad amarilla del corresponde a los modelos de simulación de tiempo continuo, la otra mitad a la simulación en tiempo discreto. De *tiempo continuo*, quiere decir que el modelo permite que los estados del sistema cambien en cualquier momento.

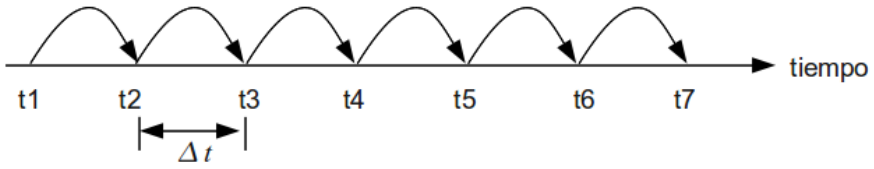


Figura 2.3: Simulación *fixed-increment*

De *tiempo discreto*, quiere decir que los cambios de estado del sistema se dan en momentos discretos de tiempo.

Un modelo *estocástico* , es el que tiene una o más variables aleatorias, al contrario del *determinista* que ante unas entradas fijas al sistema, va a producir unas salidas fijas.

Un modelo de simulación *estático* es una representación del estado sistema en un punto del tiempo. Un ejemplo de simulaciones estáticas son los modelos de Monte Carlo. Los modelos de simulación *dinámicos* representan a un sistema que evoluciona a travs del tiempo.

Los modelos de simulación que se usan en redes son *discretos, dinámicos y estocásticos,* y normalmente son llamados como modelos de simulación de evento discreto o por su siglas en ingles *DES–Discrete-Event Simulation.*

2.1.4. Simulación de Evento Discreto

La simulación de Evento Discreto *(DES)* hace un modelado de un sistema que evoluciona a través del tiempo, es decir que, las variables de estado cambian instantáneamente en puntos separados de tiempo. Solo hay estos puntos cuando ocurre un evento, considerando un evento como una ocurrencia instantánea que cambia el estado del sistema. [1]

En la simulación DES se debe llevar rastro del valor actual del tiempo simulado, que llamaremos *reloj de simulación.* Hay dos formas de avanzar el reloj de simulación, avanzar con intervalos constantes de tiempo *(fixed-increment)* o con

[1]Definición de Law y Kelton [5]

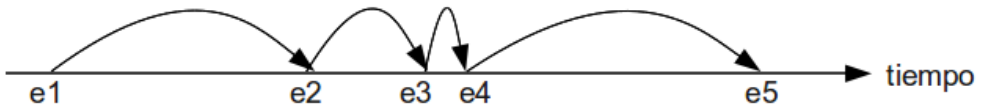


Figura 2.4: Simulación *next-event*

intervalos de tiempo basado en eventos *(next-event)*. Un ejemplo de simulación *fixed-increment o síncrona* se puede ver en la fig.2.3.Un ejemplo de simulación con intervalos basado en eventos *next-event o asíncrona* se puede ver en la fig 2.4.

Pasos para desarrollar un modelo DES

Básicamente para desarrollar un modelo de evento discreto, se siguen los siguientes pasos:

1. Determinar las metas y objetivos.
2. Construir un modelo conceptual.
3. Convertirlo en un especificación del modelo.
4. Convertirlo en un modelo computacional.
5. Verificación.
6. Validación.

En primer punto se define lo que queremos modelar.
El modelo conceptual, es lo mismo que un modelo mental ,depende de nuestro punto de vista, suele ser incompletos y no tener un enunciado preciso, no son fácilmente transmitibles. Un ejemplo de modelo conceptual es una mapa conceptual, donde se escriben ideas que para el autor tienen un orden, pero que si mira otra persona no encontrará la interconexión entre ellas.
La especificación del modelo involucra ecuaciones, pseudocodigo, y se definen las

entradas del sistema. El modelo computacional es el modelo especificado implementado en un programa de computador, que puede ser hecho con un lenguaje de propósito general o en simulador.

Verificación Vs. Validación

En la verificación se mira si el modelo computacional es acorde a la especificación del modelo. Tiene en cuenta si el modelo está bien implementado.

En la validación se mira si el modelo computacional es consistente con el sistema que se está analizando, es decir, se comprueba si se lograron los resultados esperados y si el error de precisión es acorde a los límites planteados.

Componentes del modelo DES

Aunque la gran mayoria de programas de simulación funcionan distinto, la gran mayoria tienen en cuenta los siguientes componentes :

- Estado del Sistema: Es la colección de variables de estado necesarias para describir el sistema en un punto de tiempo.
- Reloj de Simulación: Una variable que lleva el valor del tiempo simulado.
- Lista de eventos: La lista contiene el tiempo del próximo evento.
- Contador estadístico: Variables usadas para almacenar la información estadística del sistema.
- Rutina de inicialización: Inicializa el modelo de simulación en tiempo cero.
- Rutina de temporización: Determina el proximo evento y adelanta el reloj simulacion al tiempo en que el evento ocurre.
- Rutina de eventos: Actualiza el sistema cuando ocurre un evento.
- Generador de reportes: Computa un estimado de las medidas deseadas de rendimiento y produce un reporte cuando la simulación termina.

2.2. Redes de interconexión

Una red de interconexión es un sistema físico compuesto por una serie de elementos que se comportan e interrelación entre ellos de manera especifica y que sirven para facilitar la comunicación de los nodos de computo[6].

2.2.1. Técnicas de conmutación

La técnica con conmutación determina la manera en que el nodo hace el traspaso del paquete entrante. Las técnicas mas usadas son *Store and Forward y VCT.*

Conmutación Store and Forward

Al utilizar esta técnica se recibe y almacena todo el paquete*(cabecera + payload)* antes de ser enviado al siguiente nodo. Tiene varios inconvenientes respecto a la latencia y al tamaño de los buffers requerido para usar esta técnica.

Conmutación VCT

En esta técnica solo se almacena en el buffer de entrada la cabecera del paquete. El resto del paquete es trasmitido directamente después que se toma la decisión de encaminamiento. De esta forma se reduce la latencia en comparación al *Store and Forward* pero se pierde confiabilidad.

2.2.2. Topologías de red

La topología de red tiene que ver con la forma en que están interconectados los nodos. La redes se pueden clasificar según su topología principalmente en tres grupos, *redes de medio compartido, redes directas y redes indirectas.*

Redes de medio compartido

Tienen un elemento de interconexión único que no permite comunicaciones concurrentes.

Un ejemplo de este tipo de redes es la conexión *bus*, ilustrado en la figura 2.5.

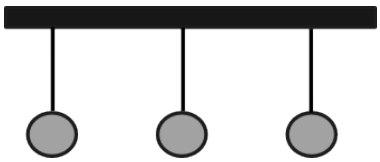


Figura 2.5: Red de medio compartido *Bus*

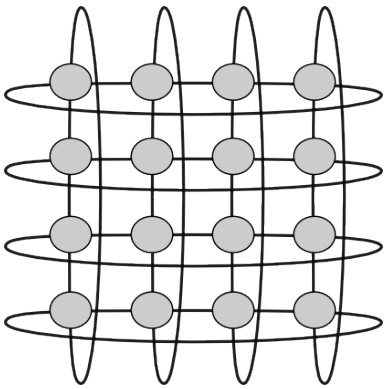


Figura 2.6: Toro 2-D de $4x4$

Redes directas

La redes directas o redes punto a punto son escalables a un gran numero de nodos [3].

En estas redes los nodos esta interconectados directamente a un pequeño numero de otros nodos. Un ejemplo de este tipo de redes son las *toro 2-D* que se ilustra en la figura 2.6.

La red toro 2-D será usada en la experimentación de los modelos que se plantean en este trabajo.

Redes indirectas

A diferencia de las redes directas, los nodos no están conectados directamente, sino a través de *switches*. En este grupo se encuentran las redes *Crossbar* y la redes *'Multistage Interconnection'(MINs)* [3, Cap.1].

2.3. Modelos de redes de interconexión

La necesidad de abstraer el funcionamiento de las redes de interconexión, lleva a la creación de modelos que tienen como objetivo hallar la latencia del sistema.

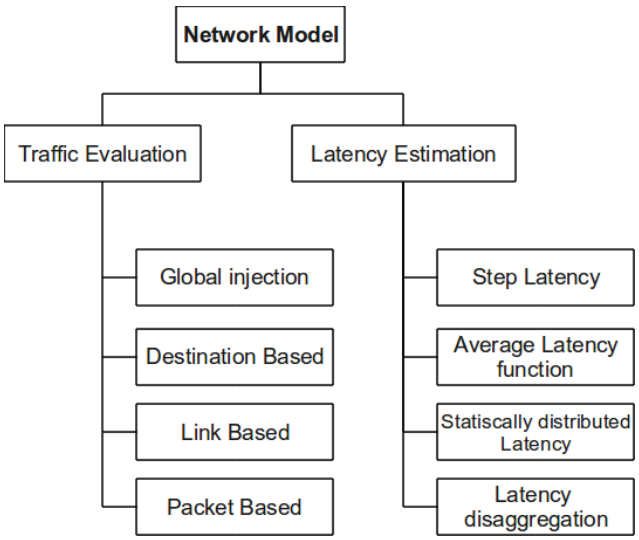


Figura 2.7: Espacio de diseño de los modelos de red

Según lo visto en la sección 2.1, los modelos son representaciones del sistema, no tienen que ser exactamente como el sistema real. Hay varios modelos desarrollados en [6] que tienen en cuenta algunas características del sistema. Ahora teniendo en cuenta las redes de interconexión, el objetivo del modelo de red es evaluar el comportamiento de la red durante un intervalo de tiempo especificado por el usuario.

Hay dos partes importantes en el espacio de diseño del modelo de red (figura 2.7). Una de éstas tiene en cuenta la evaluación del tráfico *Global injection, Destination based, Link based y Packet based* y la otra parte la estimación de la latencia. Los modelos de redes de interconexión varían en complejidad y también en el tiempo de simulación.

2.3.1. Modelo Global-Step

Este modelo se ilustra en la figura 2.8. Usa la evaluación de tráfico *Global injection* y la estimación de latencia *Step*. Aquí solo se tienen en cuenta dos valores de latencia, por lo que lo hace un modelo bastante sencillo e inexacto, pero sirve para probar la funcionalidad del hardware y software en entornos *(Full System Simulators–FSS)* [9].

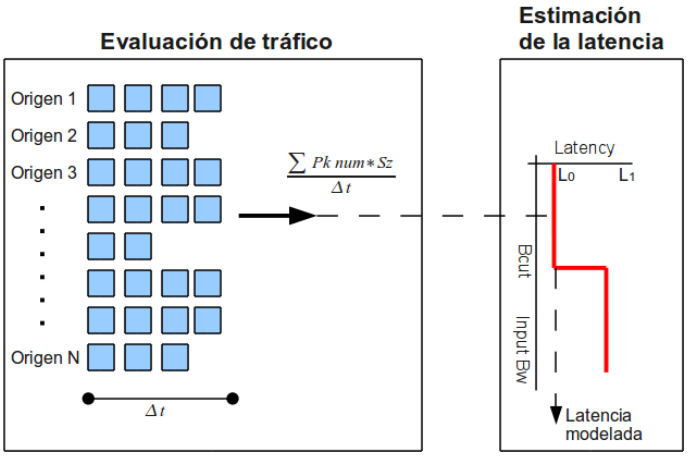


Figura 2.8: Modelo de red Global-Step

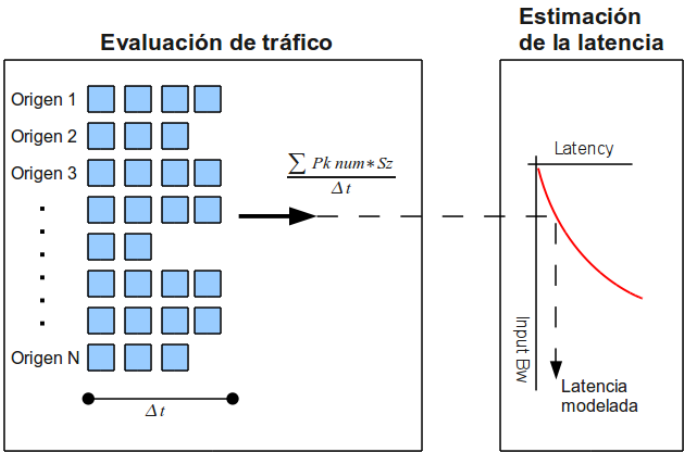


Figura 2.9: Modelo de red Global-Average

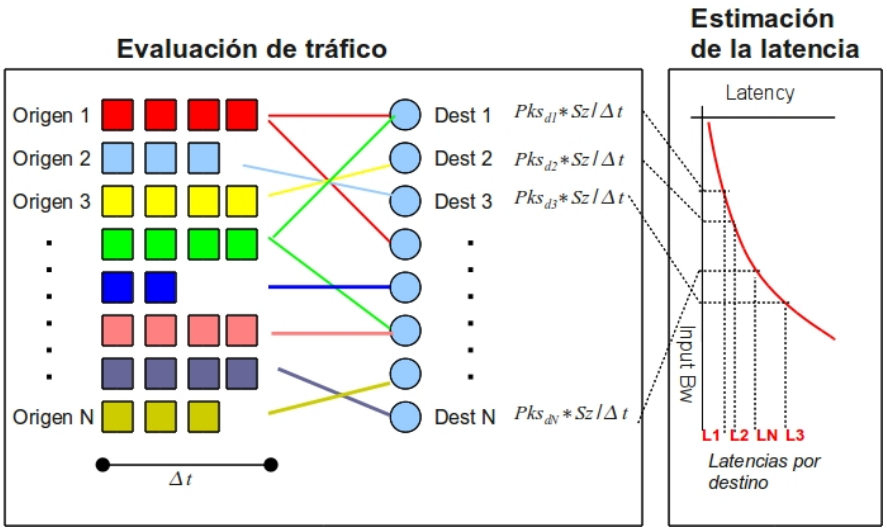


Figura 2.10: Modelo de red basado en destino

2.3.2. Modelo Global-Average

Este modelo se ilustra en la figura 2.9. Como se puede ver es una mejora con respecto al Global-Step, ya que usa un patrón mas refinado para la estimación de la latencia *(Average)*, aunque todavía en este modelo se trata a la red como una caja negra. Por lo tanto, es necesario mirar más a fondo el comportamiento de la red, para mejorar la precisión del sistema.

2.3.3. Modelo basado en destino

Mejorando los dos modelos anteriores y dejando de considerar a la red de interconexión como una caja negra, en este modelo se hace una clasificación de los paquetes por destino. Esto permite cuantificar el comportamiento de cada pareja origen–destino en la red.

Como podemos ver en la figura 2.10, el tráfico es clasificado *por destino* y se calculan las latencias por separado a cada uno de estos destinos.

$$Latency = f_{Netcomp}(N) + \phi_{QueueDynamics} \tag{2.1}$$

Para la segunda parte, la estimación de la latencia se hace con la ecuación 2.1, que tiene en cuenta la topología, el retardo de los componentes de red y la interacción dinámica de los paquetes.

- $f_{Netcomp}$ es el retardo físico, el cual es la mínima latencia acumulada por

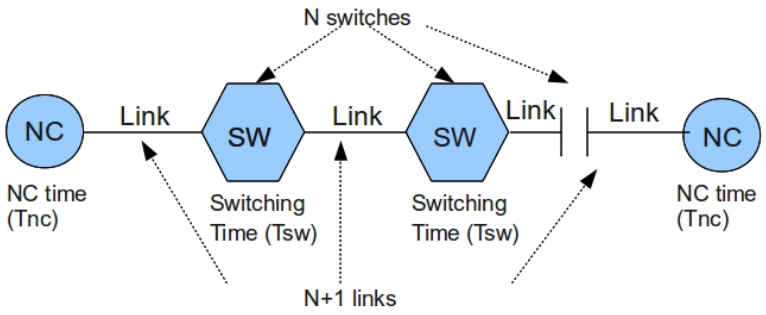


Figura 2.11: Camino origen-destino

el paquete, es decir, es la latencia cuando no hay interacción con otros paquetes.

- $\phi_{QueueDynamics}$ es el retardo de contención, que se origina cuando los recursos de red son compartidos por varios nodos origen.

Para calcular la latencia física se deben analizar los componentes que están en el camino tomado por el paquete *source-destination path* (Figura 2.11), y también se debe tener en cuenta la técnica de conmutación empleada.

$$f(N) = T_{NC} + F_t + \frac{Pk_{SZ}}{Lk_{BW}} + N * \left(F_t + T_{SW} + \frac{Pk_{HR}}{Lk_{BW}} \right) \tag{2.2}$$

Donde,

Pk_{SZ}	: Packet full size[bits]
Pk_{HR}	: Packet header size[bits]
Lk_{BW}	: Link bandwidth[bit/sec]
Ft	: Link Fly time[sec]
T_{SW}	: Switching time[sec]
T_{NC}	: Network card time[sec]
N	: Number of switches in the src-dst path

Para la técnica de conmutación *Virtual Cut-Through (VCT)* la latencia f'sica puede ser calculada [3] con la ecuación 2.2.

El retardo de contención $\phi_{QueueDynamics}$ representa la interacción dinámica de los paquetes, en la que se tiene en cuenta el patrón de tráfico, la topología, la asignación de tareas y el número de componentes compartidos[6].

El *Full System Simulator(FSS)* [9] utiliza un intervalo de tiempo *quantum* para sincronizar los diferentes nodos virtuales simulados, por lo tanto, en el modelo basado en destino también se usa, porque es pensado en interactuar con el *FSS*.

Dentro del *quantum* los paquetes llegan sin ninguna sincronización, pero para el modelo basado en destino se va a asumir que tienen una distribución de arribo *uniforme*.

El proceso para el cálculo de la contención se hará en el capítulo de *Análisis y diseño*, ya que para entender como es el procedimiento se requiere un análisis más a fondo del modelo basado en destino.

2.3.4. Modelo basado en enlace

Éste modelo es el objetivo de estudio de este trabajo, por lo que se va a tratar a más profundidad en el capítulo de *Análisis y diseño*. En pocas palabras, este modelo hace una clasificación de los paquetes según los enlaces por los que pasa, por lo que se requiere tener información del enrutamiento y de la topología de la red. Con esto se espera tener más precisión en el cálculo de la latencia.

Capítulo 3

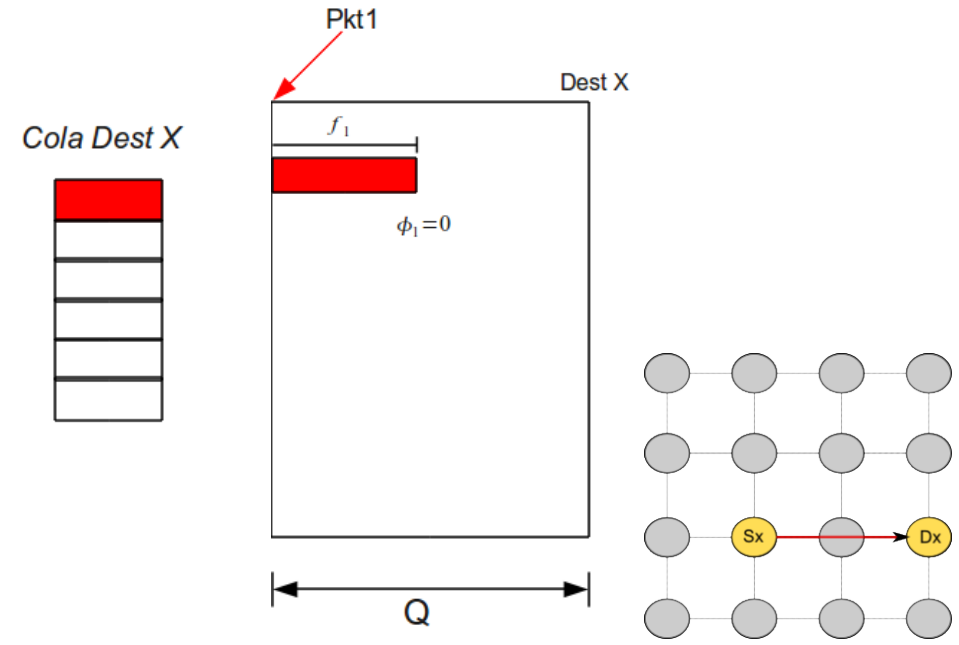
Análisis y Diseño

Como se había mencionado en el anterior capítulo, es necesario disminuir el error de precisión y al mismo tiempo aumentar la información que se tiene de la red de interconexión. En el modelo basado en destino se debe conocer la información de destino del paquete, es decir, que se debe mirar en la cabecera de los paquetes. Para el modelo basado en enlace, se debe profundizar aún más en el conocimiento que se tiene de la red de interconexión porque se debe tener la información de topología y de enrutamiento.
La topología en este caso no quiere decir que tipo de red se va a usar y de que tamao (ej: Toro 4x4), sino la información de interconexión de todos los nodos, por decir un ejemplo, los nodos 5 y 6 están interconectados por tal enlace x, y los nodos 1 y 2, se conectan por el enlace y. La información enrutamiento no es simplemente que tipo de enrutamiento se va a usar (ej. DOR, DRB, etc) sino a través de que enlace se puede desde un nodo origen alcanzar un nodo destino.

3.1. Análisis del modelo basado en destino

Como se había mencionado en la sección 2.3.3, la latencia es el resultado de la suma de la latencia física y la latencia de contención.
$Latencia = f(N) + \phi$
El proceso para hallar la contención es más complejo, está dado por un procedimiento recurrente mostrado en [8] y que se va a explicar con detalle en esta sección.

Para el ejemplo, vamos a suponer que al principio llega un solo paquete, que tiene como origen un nodo S_X y destino un nodo D_X, ver fig. 3.1. La trayecto-

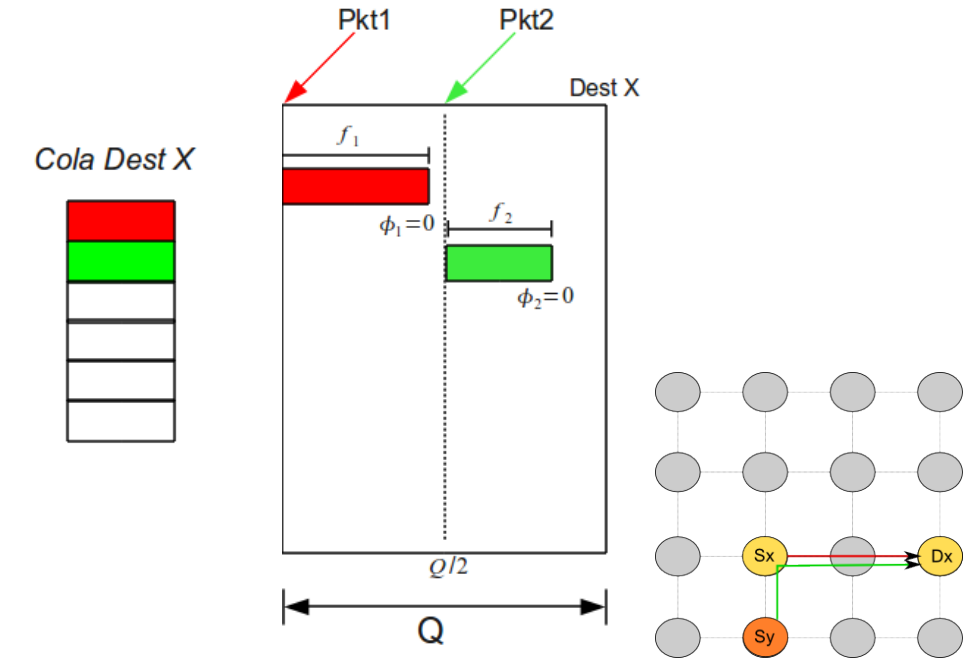


(a) Cola con 1 Pkt (b) Representación de 3 Pkts (c) Arribo de 1 Pkt

Figura 3.1: Red con 1 paquete

ria se puede ver en la parte derecha de figura 3.1c, el paquete tiene dos saltos pero para este modelo será transparente. La información de su retardo físico f_1 es almacenada en la cola que vemos en la parte izquierda de la figura 3.1a. En esta cola el espacio en rojo es que tiene la información con el tiempo de retardo fisico $f1$ y los espacios en blanco son posiciones disponibles de memoria, es decir, espacios vacos. En la parte central de la figura 3.1b, se tiene la representación en tiempo, de la cola.
Al facilitar la visualización con lo anterior, se puede observar que no hay contención porque el retardo físico f_1 del paquete no supera el tiempo del intervalo de la distribución uniforme, que es $Q/N_{Pkts} = Q$ para este caso.

Cuando llega el segundo paquete con origen S_Y y destino D_X, cuya trayectoria se puede ver en la parte derecha de la figura 3.2. Visualizando la figura, éste paquete tiene tres saltos entre el origen y destino, pero no se cuenta con esta información dentro del modelo, por que no se conoce ni la topología ni el enrutamiento. Al llegar el segundo paquete, se agrega el valor de su retardo físico $f2$ al segundo espacio de la cola, que en la parte izquierda de la figura 3.2a, se ilustra con color verde. Ahora la cola tiene en el primer espacio $f1$ y en el segundo



(a) Cola con 2 Pkts (b) Representación de 2 Pkts (c) Arribo de 2 Pkts

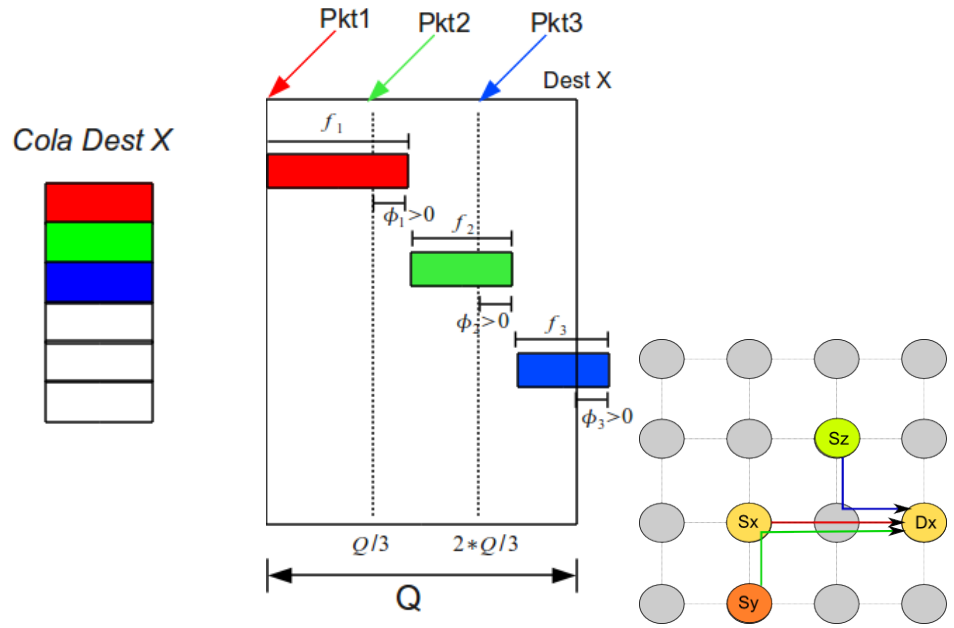
Figura 3.2: Red con 2 paquetes

espacio a $f2$.

Si observamos la parte central de figura 3.2b podemos observar la representación de la cola en tiempo, en la cual el tiempo máximo es Q, que es tiempo del *quantum*. Al arribar el segundo paquete, el tamaño de la distribución se reduce a $\frac{Q}{2}$, ver fig. 3.2b, pero este valor es aún mayor que el retardo fisico $f1$ del paquete 1, en otros términos, $f_1 < \frac{Q}{2}$, y en este caso, el retardo físico f_2 del segundo paquete también es menor que el tiempo de la distribución.

Entre más paquetes arriben dentro de un *quantum* mayor será la probabilidad de que exista contención. Como se puede puede ver en el ejemplo de la figura 3.3 , en este momento llega un tercer paquete con origen S_Z y destino D_X, y es almacenado en la cola. Éste paquete por facilidad de visualización se ilustra con color azul. Ahora la cola tiene tres espacios llenos con $f1$,$f2$ y $f3$, que tienen la información de retardo físico de los paquetes 1, 2 y 3, respectivamente. Observando la parte central de la fig. 3.3b, se puede ver la representación en tiempo de la cola del destino X.

Viendo ésta figura, Se puede notar que existe contención porque el retardo físico $f1$ del primer paquete es mayor que el tamaño de la distribución, es decir, $f_1 > \frac{Q}{3}$. Esta contención la vamos a representar como ϕ_1.



(a) Cola con 3 Pkts (b) Representación de 3 Pkts (c) Arribo de 3 Pkts

Figura 3.3: Red con 3 paquetes

Debido a que el destino solo puede recibir un paquete al tiempo, el segundo paquete debe empezar en $\frac{Q}{3}+\phi_1$, lo que provoca que haya una segunda contención ϕ_2, porque el segundo paquete sobrepasa el umbral de tiempo $\frac{2*Q}{3}$. Para hallar ϕ_2 y ϕ_3 se procede de la siguiente manera:

$$\begin{aligned} \phi_1 &= 0 + f_1 - \tfrac{Q}{3} \\ \phi_2 &= \phi_1 + f_2 - \tfrac{Q}{3} \\ \phi_3 &= \phi_2 + f_3 - \tfrac{Q}{3} \end{aligned} \tag{3.1}$$

Una vez se tienen todas las contenciones se puede hallar la latencia para el tercer paquete, que es $Lat_{Pkt3} = f_3+\phi_3$, la cual sería la latencia para este destino. En general, el cálculo de la latencia para cada arribo de paquete en un *quantum* se realiza mediante el siguiente procedimiento.

Procedimiento del modelo Dbased:

—Identificar el nodo destino.
—Calcular la distribución de arribo incluyendo el paquete nuevo.
—Calcular la contención ϕ para la cola de destino correspondiente

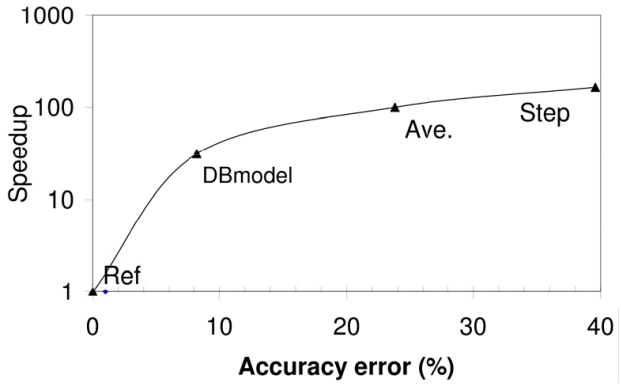


Figura 3.4: Compromiso entre precisión y velocidad de simulación

$$\phi = \begin{cases} \phi[i-1] = \phi[i-2] + f[i-1] - \frac{Q}{N_{Pkts}} & \forall i > 0, \text{y } \phi > 0 \\ \phi[i-1] = 0 & \forall \phi < 0 \end{cases} \quad (3.2)$$

—Calcular la latencia para el i–ésimo paquete.
$Lat[i] = f_{Netcomp}(N)[i] + \phi_{QueueDynamics}[i-1]$

Donde,

$f[i-1]$	: Retardo físico del paquete $i-1$ con destino x.
$\phi[i-1]$	: Retardo de contención del paquete $i-1$ con destino x.
Q	: Valor del *quantum*.
N_{Pkts}	: Número de paquetes procesados dentro del *quantum*.

En caso que llegara un cuarto paquete se tendrían que recalcular todas las contenciones de nuevo porque se disminuiría el tiempo de la distribución a $\frac{Q}{4}$. El valor de contención ϕ_4 se calcularía de la siguiente manera:

$$\begin{aligned} \phi_1 &= 0 + f_1 - \tfrac{Q}{4} \\ \phi_2 &= \phi_1 + f_2 - \tfrac{Q}{4} \\ \phi_3 &= \phi_2 + f_3 - \tfrac{Q}{4} \\ \phi_4 &= \phi_2 + f_3 - \tfrac{Q}{4} \end{aligned}$$

Según las pruebas hechas en [6], el modelo basado en enlace *DB Model*, es alrededor de 30 veces más rápido que el modelo de referencia y solo tiene una pérdida de prestaciones $\approx 8\,\%$.

Los modelos *Global-Step y Global-Average* son más rápidos pero pierden mucha precisión, ver figura 3.4 [1]. En el modelo *Global-Average* se logra una velocidad *100x*, con un error de precisión de hasta el 24 % y en el modelo *Global-Step* se logra una velocidad *165x*, con un error de precisión de hasta el 39 %.

Buscando un compromiso de velocidad y bajo error de precisión, se ve claramente el objetivo que se quiere lograr al diseñar el modelo basado en enlace *LBModel*, el cúal es disminuir el error, debido a que se tiene más información de la red. Aunque se pierda velocidad en el modelo, ésta no puede llegar hasta la velocidad del modelo de referencia.

3.2. Diseño del modelo basado en enlace

Con el objetivo de reducir el error de precisión al hallar la latencia general, se debe hacer un modelo más especifico que tenga en cuenta cuando dos o más paquetes estén usando un mismo enlace, lo cual daría una aproximación mayor al sistema real.
Así como el sistema basado en destino *(DBModel)* hacía una clasificación de los paquetes entrantes teniendo en cuenta el *destino*, el sistema basado en enlace hace una clasificación de los paquetes entrantes teniendo en cuenta el *path* [2] que utilizan. Pero como en la información que traen los paquetes los paquetes *(Source, Destination)* no hay información del *path*, es necesario hallarlo, con la información de enrutamiento y topología de nuestra red. Ésta información hasta ahora no se había tenido en cuenta.

La red de interconexión que se usará es la *toro 2-D* de *4x4*, al nodo inferior izquierdo se le llamará *Nodo 0* y al superior derecho *Nodo 15*.

3.2.1. Procedimiento para hallar el *listado de enlaces*

Para hallar el *path* se requiere tener información adicional de nuestra red de interconexión. En el *LBModel* ahora tendremos en cuenta la información de topología y la información de enrutamiento. Ésta información la tendremos que almacenar en dos tablas, una para enrutamiento y otra para topología.

[1] Figura de la pag.70 de [6]
[2] Trayectoria del paquete o, información de los nodos o links por los que pasa

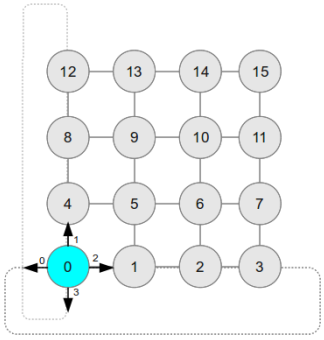


Figura 3.5: Enlaces del nodo 0 en red toro 4x4

Tabla de topología

La tabla de topología tiene la información de los enlaces de cada nodo y cúal nodo esta conectado por cada enlace. Cuando hacemos una lectura de de esta tabla se tiene como entradas el enlace y el nodo origen, y lo que queremos saber es el nodo destino para el respectivo enlace.

$$F(Nodo_{origen}, enlace) \rightarrow Nodo_{destino} \tag{3.3}$$

Como se ve en la ecuación 3.3 las dos entradas o datos que conocemos son el *nodo origen* y el *enlace*, y lo que nos retorna es el *nodo destino* de la tabla.

Tomando como ejemplo el nodo 0, ver figura 3.5, los nodos vecinos son 3, 4, 1, y 12. El nodo 3 está conectado a través del enlace 0, el nodo 4 a través del enlace 1, el nodo 1 a través del enlace 2, y el nodo 12 a través del enlace 3.

La tabla de topología para el nodo 0 quedaría como sigue:

Nodo Origen	Enlace	Nodo Destino
0	0	3
0	1	4
0	2	1
0	3	12

Como es necesario que la tabla contenga la información de los enlaces de todos los nodos de la red, si la mostramos como lo acabamos de hacer sería necesario una tabla de $3x(4xN)$ donde N es el numero de nodos. Si tenemos 16 nodos, la tabla sería de $3x64$, por lo tanto, haría falta mostrar la tabla de topología de una manera más eficiente.

Una manera mejor de mostrar la tabla de topologia es como sigue:

Enlace \ Origen	0	1	...	15
0	3	0	...	14
1	4	5	...	3
2	1	2	...	12
3	12	13	...	11

Ahora el tamaño de la tabla es de $4xN$, que para 16 nodos sería de $4x16$, con lo que nos daría un tamaño menor que en la tabla que habíamos planteado en un principio. Si miramos la columna Origen 0 y la Fila 0, el valor es 3, en términos de la ecuación 3.3 seria $F(0,0) \rightarrow 3$.

Tabla de enrutamiento

La tabla de enrutamiento tiene la información del enlace que debe tomar el paquete para ir a un destino dado. Cuando se hace una lectura de la tabla se tienen el nodo origen y el nodo destino, y se retorna el enlace.

$$F(Nodo_{origen}, Nodo_{destino}) \rightarrow enlace \tag{3.4}$$

Observando la ecuación 3.4 tenemos como entradas el *nodo origen* y el *nodo destino*, y la tabla nos da el *enlace*.

El enrutamiento que se usará para este modelo va a ser el *enrutamiento X–Y*. El paquete primero debe usar el eje X y cuando llega a la misma columna del nodo destino, empieza a usar el eje Y. Para una malla 2D el algoritmo se muestra en el libro [3, cap.1], para un toro 2D cambia porque se debe tener en cuenta el enlace que interconecta a los extremos. Para el toro 2D usó el siguiente procedimiento:

1. **Entradas:** Coordenadas del nodo actual (X_{actual}, Y_{actual}) y coordenadas del nodo destino $(X_{destino}, Y_{destino})$
   Salida: Enlace a usar.

2. $X_{offset} = X_{destino} - X_{actual}$
   $Y_{offset} = Y_{destino} - Y_{actual}$

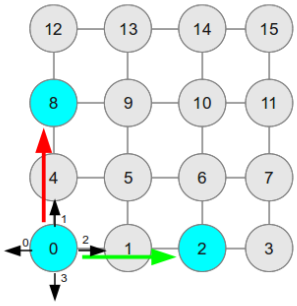


Figura 3.6: Routing desde nodo 0

3. **if** $X_{offset} > 0$ **and** $X_{offset} < \frac{N}{2}$
   then *Salida:=enlace 2*
   endif
   if $X_{offset} > 0$ **and** $X_{offset} > \frac{N}{2}$
   then *Salida:=enlace 0*
   endif

4. Cuando $X_{offset} = 0$
   if $Y_{offset} > 0$ **and** $Y_{offset} < \frac{N}{2}$
   then *Salida:=enlace 1*
   else *Salida:=enlace 3*
   endif

Ahora miraremos un ejemplo en el que el nodo 0 envia un paquete al nodo 2 y al nodo 8, ésto es para ilustrar como se llenaría la tabla de enrutamiento. Mirando la figura 3.6, el *nodo 0* utiliza el *enlace 2* si el paquete va dirigido hacia el *nodo 2*, y utiliza el *enlace 1* si el paquete va dirigido hacia el *nodo 8*. En términos de la ecuación 3.4, $F(0, 2) \rightarrow 2$ y $F(0, 8) \rightarrow 1$.

La tabla de enrutamiento para el *nodo 0* quedaría como sigue:

Origen	Destino	Enlace
0	1	2
0	2	2
⋮	⋮	⋮
0	8	1
⋮	⋮	⋮
0	15	0

Como es necesario que la tabla contenga la información de enrutamiento de todos los nodos de la red, si la mostramos como lo acabamos de hacer sería necesario una tabla de $3x(NxN)$ donde N es el numero de nodos. Si tenemos 16 nodos, la tabla sería de $3x256$, por lo tanto, haría falta mostrar la tabla de topología de una manera más eficiente.
Una manera mejor de mostrar la tabla de topologia es como sigue:

Origen \ Destino	0	1	2	...	15
0	—	2	2	...	0
1	0	—	2	...	2
2	0	0	—	...	2
⋮	⋮	⋮	⋮	⋱	⋮
15	2	0	0	...	—

Ahora el tamaño de la tabla es de NxN, que para 16 nodos sería de $16x16$, con lo que nos daría un tamaño menor que en la tabla que habíamos planteado en un principio. En la tabla, las casillas que tienen un guión se debe a que no se considera un enlace de *loopback*, es decir, si el origen y destino es el mismo, no hay un enlace para este caso.

Listado de enlaces

Ahora que sabemos como se hallaron las tablas de enrutamiento y topología, vamos a hallar el listado de enlaces *(links)* usando estas dos tablas.
El procedimiento para hallar el listado de links es el siguiente:

1. **Entradas:** Nodo origen (src) y nodo destino (dst)

2. Chequear *routing table[src][dst]* → link
3. Agregar el link al link-list
4. Chequear *topology table[link][dst]* → src-temp
5. Iterar hasta que src=src-temp;
6. **Output:**=link-list

En el listado de links, se usa la estructura *list* del *Standard Template Library–STL* de C++ [1], y se usa por su eficiencia.

3.2.2. Implementación del modelo Lbased

Sabiendo el listado de enlaces por los que pasa el paquete, se puede ahora crear una cola por enlace que tenga la información de la latencia física para el paquete en vacío.

$$Pkt(src, dst) \rightarrow PktN_{links} = (Link_1, Link_2, \ldots, Link_N) \tag{3.5}$$

Mirando la ecuación 3.5, cuando llegue un paquete que trae información de origen(src) y destino(dst), se va a guardar la latencia fisica f en un espacio disponible de la cola de cada uno de los enlaces $Link_1, Link_2, \ldots, Link_N$.

Si llegan dos paquetes Pkt1 y Pkt2, que tiene latencias f1 y f2 respectivamente, el Pkt1 usa los links $(Link_1, Link_5)$ y el Pkt2 usa los links $(Link_5, Link_3)$, entonces las colas para cada uno de los links quedaria asi:

$Cola(Link_1) = [f1]$
$Cola(Link_5) = [f1; f2]$
$Cola(Link_3) = [f2]$

Contención en un enlace

Hay tres paquetes Pkt1, Pkt2 y Pkt3 que usan el mismo Link X, cada uno de estos tiene latencias físicas f1,f2 y f3 respectivamente. Si se observa detalladamente la figura 3.7 f1, f2 y f3 son almacenadas en la cola que corresponde al link X. En la figura se muestra en la parte izquierda, el diagrama de la red con la trayectoria

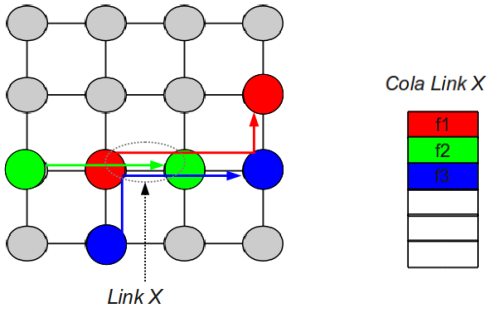


Figura 3.7: Contención en Link X

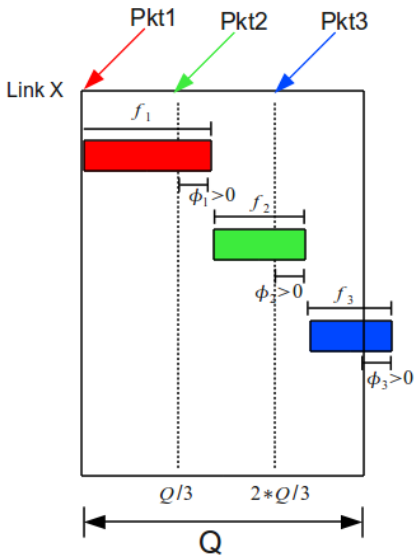


Figura 3.8: Cola de Link X vista en tiempo

de los paquetes. En el modelo basado en enlace se tendrá en cuenta la trayectoria porque se dispone de información de topología y enrutamiento. Por lo tanto, se sabe que los tres paquetes usan el mismo Link X.

Si miramos la cola en tiempo, ver figura 3.8, f1 sobrepasa el valor Q/3, por lo que existiría una contención ϕ_1, para f2 y f3 también existiría contención ϕ_2 y ϕ_3 respectivamente.
Para este Link X existirá una contención ϕ_3.

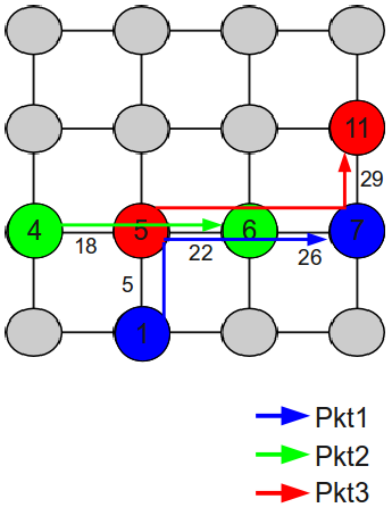


Figura 3.9: Ejemplo para LBModel en un quantum

Aproximación analítica del Modelo Lbased

Para el modelo Lbased ya conocemos el procedimiento que nos retorna la lista de enlaces.
Para poder enumerar todos los enlaces de la red vamos a considerar primero los bits correspondientes al nodo y después los bits correspondientes al enlace. Es decir el enlace 1 del nodo 2 se representaría como (001001).

$$\underbrace{0010}_{nodo}\ \underbrace{10}_{enlace} \tag{3.6}$$

En la ecuación 3.6 se muestra mejor la numeración que se hizo de los enlaces. Los primeros cuatro bits corresponden al nodo y los dos últimos al enlace. Entonces si usamos el nodo 2–link 2, nos da en bits (001010). Este enlace también se puede expresar en decimal como 10. En otros términos sería, $001010 \xrightarrow{dec} 10$.
Si por ejemplo, el nodo 5–link 1, el enlace sería (010101), que en decimal es el número 21.
Aunque parezca un poco confusa ésta numeración, se hizo para facilitar la implementación del modelo en C++.

Ahora, para ilustrar como es el procedimiento de nuestro modelo detalladamente, debemos observar la figura 3.9. En esta figura tenemos tres paquetes con distintos colores cada uno para facilitar la observacion y analisis. El paquete 1 que tiene como origen el nodo 1 y destino el nodo 7, usa los enlaces 5, 22 y 26. De

ésto se encarga la rutina de listado de links. Generalizando para los tres paquetes, usaremos como base la ecuación 3.5, para facilitar la visualización.

$$\begin{aligned} Pkt_1(1,7) &\rightarrow Pkt1_{links} = (5,22,26) \\ Pkt_2(4,6) &\rightarrow Pkt2_{links} = (18,22) \\ Pkt_3(5,11) &\rightarrow Pkt3_{links} = (22,26,29) \end{aligned} \tag{3.7}$$

La ecuación 3.7 nos permite ver los nodos origen y destino para cada paquete en la parte izquierda, y los links que usa el paquete en la parte derecha. También podemos ver que el paquete 1 tiene 3 saltos *(hops)*, el paquete 2 tiene 2 saltos y el paquete 3 tiene 3 saltos.
En total, los links usados en este ejemplo son $Links = [5, 18, 22, 26, 29]$.
Ahora seguiremos a representar en tiempo la cola de cada uno de estos links.

En la figura 3.10, podemos ver la representación en tiempo de lo que está almacenado en cada una de las colas, de los links que estamos usando para este ejemplo. Los gráficos de la parte superior corresponden a los links 5, 18 y 29, respectivamente. El link 5 solo tiene almacenado el valor de f1 que corresponde al Pkt 1, el link 18 solo tiene almacenado el valor de f2 que corresponde al Pkt2 y el link 29 solo tiene almacenado f3 que corresponde al Pkt 3. Como se había mencionado antes, los f1, f2 y f3, son las latencias físicas de los correspondientes paquetes.
Por otro lado, en la parte inferior de la figura 3.10 están las colas representadas en tiempo, de los links 26 y 22 respectivamente. En la parte izquierda de la figura esta la colas en tiempo del link 26, la cual contiene dos paquetes, el Pkt 1 y el Pkt 3. El Pkt1 no supera el umbral de contención por lo que $\phi_1 = 0$ y el Pkt3 tampoco supera el umbral por lo que $\phi_2 = 0$. Notese que a la contención del Pkt3 le llamamos ϕ_2, porque aunque es el tercer paquete en el *quantum*, para el link 26 es el segundo paquete.
En la parte derecha de la figura está representada la cola en tiempo del link 22. Dentro del *quantum* por este link pasan tres paquetes Pkt1, Pkt2 y Pkt3. Los tiempos de latencia fisica de los tres paquetes (f1, f2 y f3) rebasan su respectivo umbral de contención por lo que para el Pkt1 hay una contención $\phi_1 > 0$, para el Pkt2 hay una contención $\phi_2 > 0$ y también para el Pkt3 existe $\phi_3 > 0$.

Ahora, analizando los paquetes en función de las contenciones los podríamos

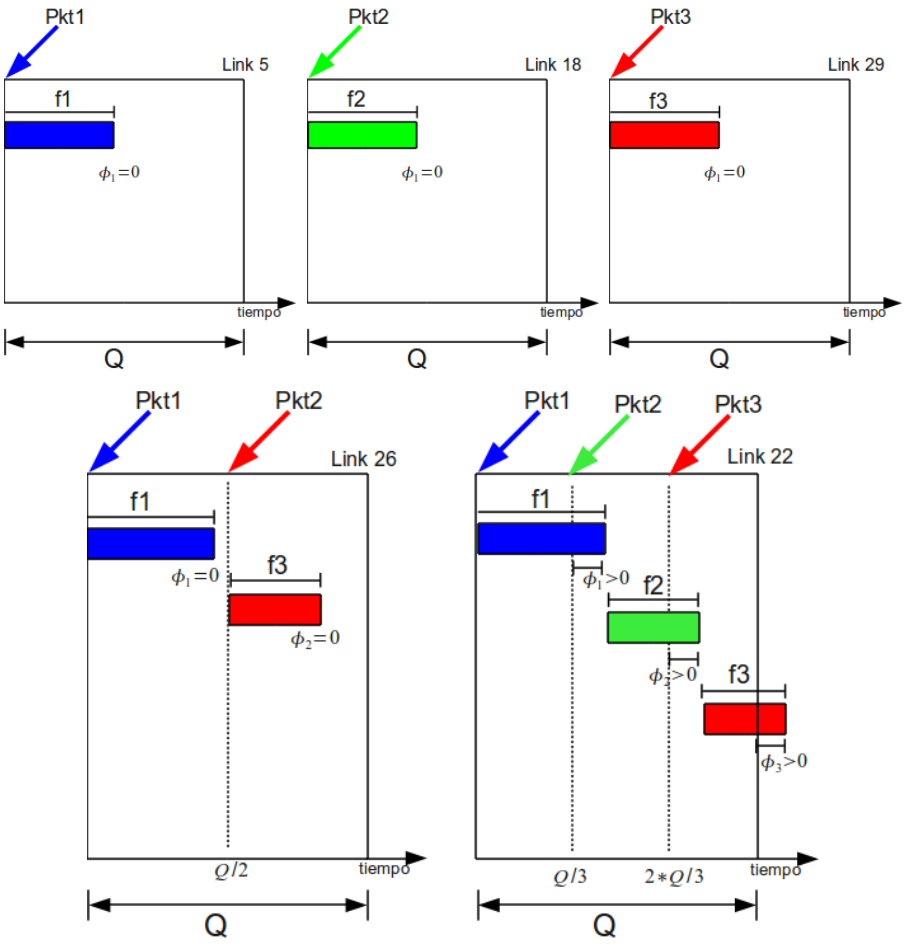


Figura 3.10: Colas de cada link vistas en tiempo

representar de la siguiente manera:

$$\begin{aligned} Pkt1_{links}(5,22,26) \rightarrow Pkt1_{contencion} &= F(0,\phi_3,0) \\ Pkt2_{links}(18,22) \rightarrow Pkt2_{contencion} &= F(0,\phi_3) \\ Pkt3_{links}(22,26,29) \rightarrow Pkt3_{contencion} &= F(\phi_3,0,0) \end{aligned} \tag{3.8}$$

Como vemos en la ecuación 3.8 los tres paquetes sufren una contención de ϕ_3 porque todos pasan el Link 22.
Si aparece un cuarto paquete por el Link 22 (por ejemplo, $src = 5$, $dest = 7$) que generara una contencion ϕ_4, esta nueva contención seria mayor que la anterior ϕ_3. Con esto los cuatro paquetes tendrían una contención ϕ_4.

Mostrándolo de una mejor manera:

$$\begin{aligned} Pkt1_{links}(5,22,26) \rightarrow Pkt1_{contencion} &= F(0,\phi_4,\phi_3{}') \\ Pkt2_{links}(18,22) \rightarrow Pkt2_{contencion} &= F(0,\phi_4) \\ Pkt3_{links}(22,26,29) \rightarrow Pkt3_{contencion} &= F(\phi_4,\phi_3{}',0) \\ Pkt4_{links}(22,26) \rightarrow Pkt3_{contencion} &= F(\phi_4,\phi_3{}') \end{aligned} \tag{3.9}$$

Como podemos apreciar en la ecuación 3.9, aparece una contención en el link 26, porque ante la llegada del cuarto paquete el umbral de contención disminuyó, por lo que las latencias físicas de cada paquete que pasa por ese link supero éste limite.
Es decir, para el link 26 va a aparecer una contención $\phi_3{}'$ que es distinta a la ϕ_3 del link 22, y que para nuestro ejemplo es menor que la contención ϕ_4 del link 22. Basándonos en este ejemplo tenemos que la contención para los cuatro paquetes será ϕ_4, es decir, que para los cuatro paquetes la función que nos permitiría llegar a este resultado será:

$$\begin{aligned} Pkt1_{contencion} &= max(0,\phi_4,\phi_3{}') \\ Pkt2_{contencion} &= max(0,\phi_4) \\ Pkt3_{contencion} &= max(\phi_4,\phi_3{}',0) \\ Pkt3_{contencion} &= max(\phi_4,\phi_3{}') \end{aligned} \tag{3.10}$$

La ecuación 3.10 será la base del modelo matemático, para determinar la contención en el link. Para hallar la contención de un paquete, siempre se tendrá en

cuenta la contención mayor entre sus links. Pero se requiere hacer un procedimiento o algoritmo para calcular la latencia del sistema.

Procedimiento del modelo Lbased:

—Identificar los links por los que pasa el paquete.
—Calcular la distribución de arribo para cada link, incluyendo el paquete nuevo.
—Calcular la contención ϕ para cada cola de link correspondiente

$$\phi = \begin{cases} \phi[i-1] = \phi[i-2] + f[i-1] - \frac{Q}{N_{Pkts}} & \forall i > 0, \text{y } \phi > 0 \\ \phi[i-1] = 0 & \forall \phi < 0 \end{cases} \tag{3.11}$$

—Calcular la contención máxima entre los links por los que pasa el paquete.
$\phi_{Pkt} = max(\phi_1, \phi_2, \cdots, \phi_k)$
—Calcular la latencia para el i–ésimo paquete.
$Lat[i] = f_{Netcomp}(N)[i] + \phi_{QueueDynamics}[i-1]$

Donde,

$f[i-1]$	: Retardo físico del paquete $i-1$ entre src-dst x.
$\phi[i-1]$	: Retardo de contención del paquete $i-1$ en el link x.
Q	: Valor del *quantum*.
N_{Pkts}	: Número de paquetes procesados dentro del *quantum*.

Capítulo 4

Experimentos

En este capítulo se va a empezar con una verificación del modelo basado en enlace *LBModel* con algunos patrones de tráfico para saber si calcula bien la contención esperada. En la segunda sección se harán una serie de experimentos para hallar el error del *LBModel* con respecto al modelo de referencia.

4.1. Verificación de *LBModel vs DBModel*

—Experimento 1: Para verificar la implementación del modelo LBModel hecha en C++ [1], se creó una traza de cuatros paquetes que no chocan (no comparten ni destino, ni enlace), por lo tanto, la latencia de la pareja origen–destino debe ser igual a la latencia física, es decir, $Latencia = f(N) + \phi$ donde $\phi = 0$.

Como se puede ver en la figura 4.2, todos los paquetes tienen la misma latencia, y al comparar los dos modelos, la latencia es igual. Con este primer experimento, se asegura que el nuevo modelo no tiene errores de implementación.

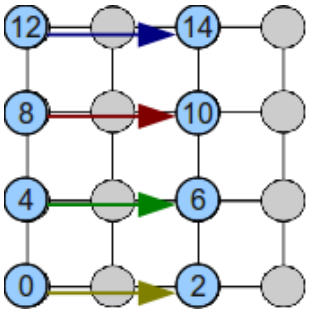


Figura 4.1: Experimento 1

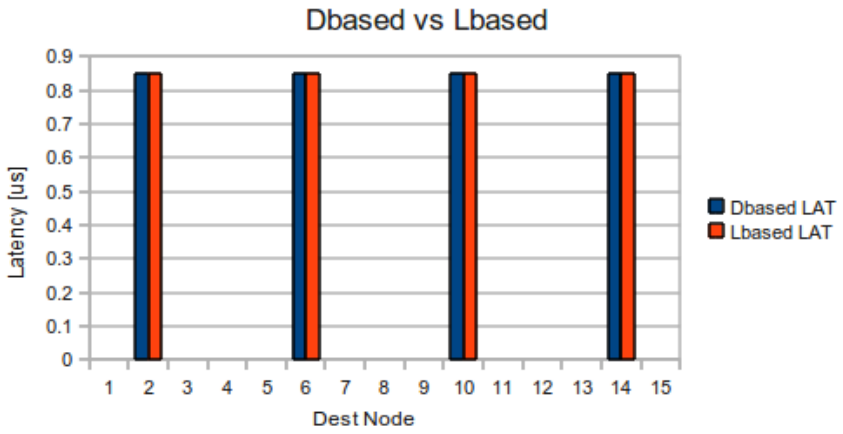


Figura 4.2: Resultados del expermiento 1

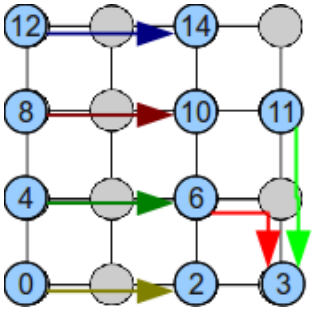


Figura 4.3: Experimento 2

—Experimento 2: En este experimento se va a utilizar una traza en donde, hay dos paquetes que tienen el mismo destino. Para este caso, se espera que los dos modelos tengan la misma contención, porque el modelo *DBModel* ve que comparten el mismo destino y el modelo *LBModel* ve que comparten el mismo enlace.

Como se puede ver en la figura 4.4, para los cuatro paquetes que tienen como destino los nodos (2, 6, 10 y 14) no tienen contención porque no chocan, pero los paquetes que tiene como destino el nodo 3, comparten el mismo destino y por lo tanto, el mismo enlace. La contención del DBModel y LBModel son iguales, porque solo se da en un punto.

—Experimento 3: En este experimento se va a utilizar una traza en donde, hay dos paquetes que usan el mismo enlace pero no comparten destino. Para este caso, el DBModel no debe detectar contención, pero sí el LBModel.

Como se ve en los resultados del experimento, ver fig. 4.6, el DBModel no

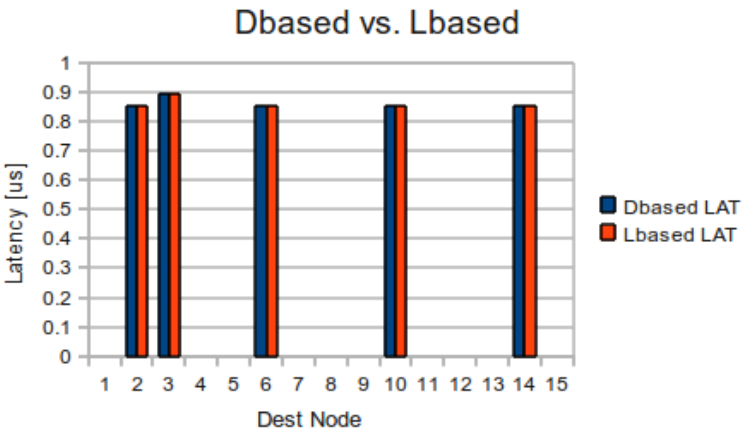


Figura 4.4: Resultados del expermiento 2

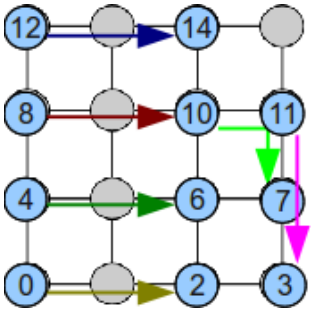


Figura 4.5: Experimento 3

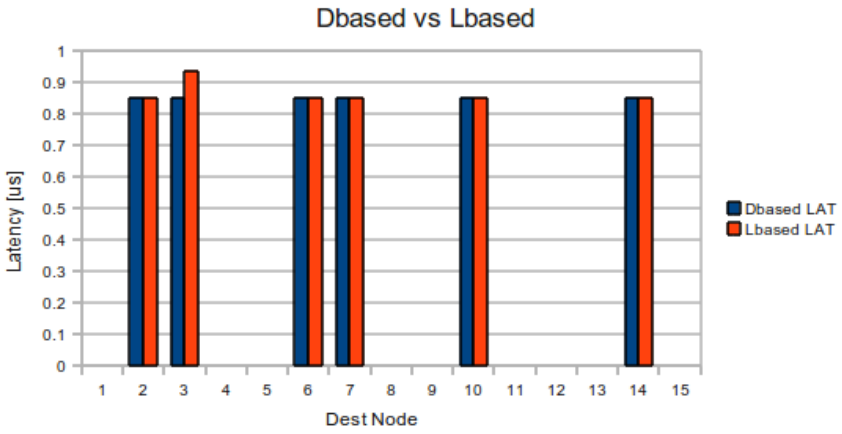


Figura 4.6: Resultados del expermiento 3

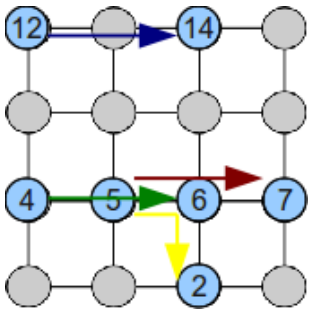


Figura 4.7: Experimento 4

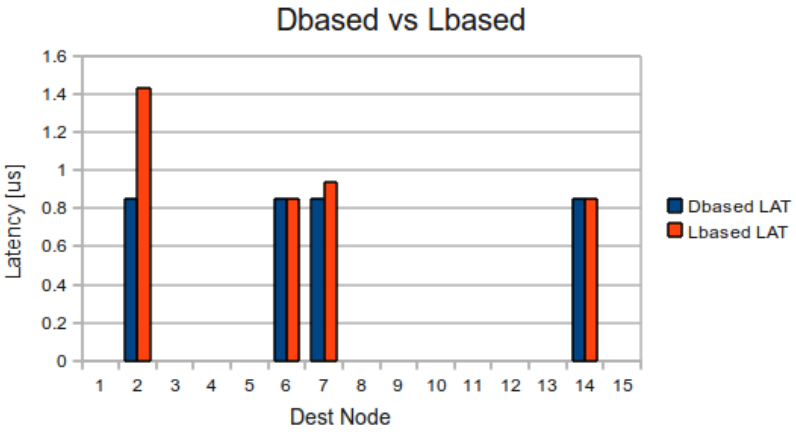


Figura 4.8: Resultados del expermiento 4

detecta la contención entre los paquetes que tienen como destino los nodos 3 y 7, es decir, $\phi = 0$.
El LBModel detecta dicha contención en el enlace entre los nodos 7 y 11, es decir, $\phi > 0$. Por eso el paquete entre los nodos 11 y 3, tiene una mayor latencia que el resto.

—Experimento 4: En este experimento se va a utilizar una traza en donde, hay tres paquetes que comparten el mismo enlace. El primer paquete de la es el que tiene como destino el nodo 6, el segundo tiene como destino el nodo 7 y el último, en el que se almacena la contención tiene como destino el nodo 2. El cuarto paquete entre los nodos 12 y 14, no tiene ninguna contención, ni con DBModel ni con LBModel.

Como se puede ver en la figura 4.7, los tres paquetes chocan en el enlace entre los nodos 5 y 6. Dado que el paquete entre los nodos 5 y 2 es el último que entra a la cola de contención, es el que tendrá la contención mayor.

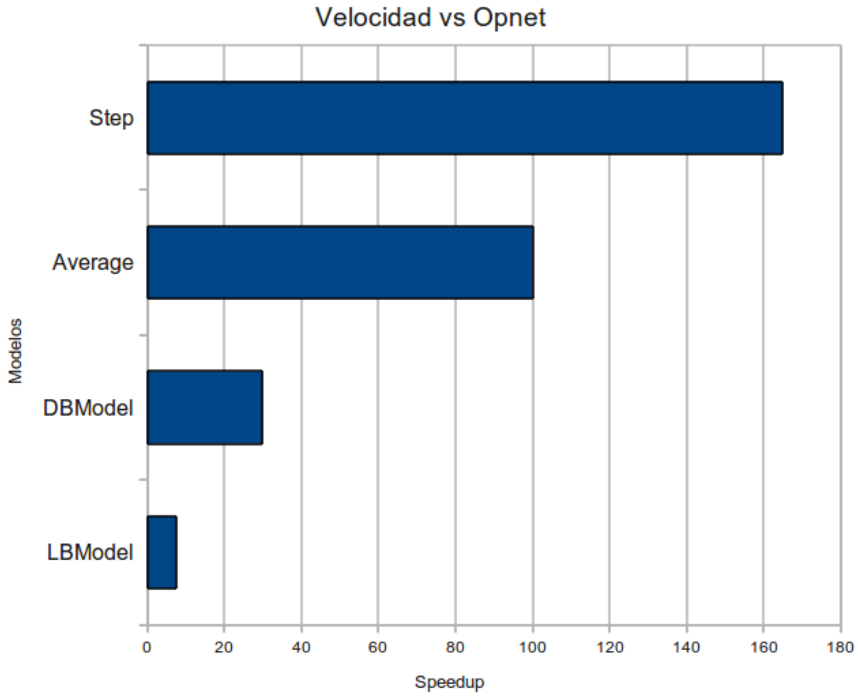


Figura 4.9: Resultados del expermiento 5

En los resultados mostrados en la figura 4.8, que el paquete hacia el destino 2 es el que experimenta mayor contención. Como este ejemplo, no es detectable por DBModel, éste modelo considera que $\phi = 0$ por lo que la latencia es igual a la latencia física.

—Experimento 5: En este experimento se utiliza una traza de 700.000 paquetes. El objetivo es medir el tiempo de ejecución del modelo de simulación Dbased y el Lbased. Considerando que el modelo Dbased es 30 veces más rápido que Opnet, comprobado en [7], se hace una extrapolación del tiempo de Lbased vs Dbased al tiempo de Lbased vs Opnet. El speedup de los modelos Step y Average, son de 165 y 100 veces más rápidos respecto a Opnet, se incluyen en la gráfica con el fin de comparar el tiempo obtenido con el modelo Lbased.

El tiempo de ejecución del modelo de simulación Lbased con los 700.000 paquetes es de 574 segundos, y el del modelo Dbased es de 147 segundos, lo que da que Dbased es 3.9 veces más rápido que el modelo Lbased. Como se puede ver en la figura 4.9, se hizo la extrapolación de que si Dbased es 30 veces más rápido que Opnet, entonces Lbased es 30/3,9. También se puede notar un decremento casi continuo entre los modelos, del modelo Average al modelo Dbased hay un

decremento de velocidad $\approx$ 3,3 y del modelo Dbased al modelo Lbased hay un decremento de 3,9.

Capítulo 5

Conclusiones

Los computadores paralelos son cada vez más rápidos y por lo tanto, requieren tener redes de interconexión más rápidas. Partiendo de que la red es un elemento crítico del sistema, requiere un estudio a profundidad para mejorar su rendimiento, por lo tanto se necesita la construcción de un modelo.
Al *modelar* una red de interconexión se pueden conocer las entidades que interactúan y su comportamiento, lo que nos deja prever como reaccionará el sistema ante posibles cambios.

En este trabajo, se han estudiado algunos modelos de redes de interconexión y se ha hecho el diseño e implementación del modelo basado en enlace *LBModel*. Por lo visto, durante el estudio de los modelos existentes, *Step, Average y DBased*, la velocidad y precisión son antagónicas, es decir, que si se aumenta en velocidad, se disminuye en precisión y viceversa.

El estudio de los modelos de red existentes permitió adquirir las herramientas de análisis necesarias para hacer en primera medida la especificación del *LBmodel*.

Mediante el estudio de casos, se pudo hacer una aproximación analítica al sistema a modelar, y en un paso posterior, permitió llegar a un modelo computacional para poder simular.

Mirando los experimentos en este trabajo, se puede concluir que la implementación del *LBModel* detecta bien los casos que se habían planteado. Cuando se compara con el modelo Dbased se nota que tienen el mismo comportamiento si

no hay choques entre los paquetes, y también si se comparte el mismo destino. Cuando los paquetes comparten el mismo enlace pero no el mismo destino, el *LBModel* hace la detección y genera una contención en el enlace. Por lo tanto, con esto se aumenta la precisión del modelo respecto a los modelos anteriormente mencionados.

El modelo basado en enlace logra una velocidad inferior al modelo Dbased pero superior a Opnet. Considerando que este modelo es más preciso porque tiene mayor información de la red, podemos concluir que el modelo basado en enlace logra un compromiso entre velocidad y precisión.

Como trabajo futuro, se podría utilizar alguna traza de un programa paralelo real que haga uso intensivo de la red de interconexión y comparar los tiempos de ejecución y precisión de los modelos basados en enlace y basados en destino.

Bibliografía

[1] Manual de referencia c++, libreria stl, jun 2010. http://www.cplusplus.com, 2010.

[2] Top 500 supercomputers site, interconnection family, jun 2010. http://www.top500.org, 2010.

[3] J. Duato, S. Yalamanchili, y L.M. Ni. *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003.

[4] D.R. Insúa, J.M. Jiménez, et al. *Simulación: métodos y aplicaciones*. México, DF:. Alfaomega;, 2009.

[5] A.M. Law y W.D. Kelton. *Simulation modeling and analysis*. McGraw-Hill New York, 1991.

[6] D. Lugones. *Políticas de encaminamiento multicamino en redes de interconexión de altas prestaciones*. Tesis Doctoral, CAOS - UAB, 2009.

[7] D. Lugones, D. Franco, E. Argollo, y E. Luque. Models for high-speed interconnection networks performance analysis. En *MASCOTS*, págs. 1–4. IEEE, 2009. URL http://dx.doi.org/10.1109/MASCOT.2009.5366358.

[8] D. Lugones, D. Franco, D. Rexachs, J.C. Moure, E. Luque, E. Argollo, A. Falcón, D. Ortega, y P. Faraboschi. High-speed network modeling for full system simulation. En *IISWC*, págs. 24–33. IEEE, 2009. ISBN 978-1-4244-5156-2. URL http://doi.ieeecomputersociety.org/10.1109/IISWC.2009.5306799.

[9] D. Lugones, E. Luque, D. Franco, J.C. Moure, D. Rexachs, P. Faraboschi, D. Ortega, G. Giménez, y A. Falcón. Initial studies of networking simulation on cotson. 2009.

[10] Opnet Technologies. Opnet modeler accelerating network r&d, june 2008. http://opnet.com, 2008.